



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/660,353	09/11/2003	P. Anders I. Bertelrud	2095.001100/P3126	5128
62293	7590	12/15/2009		
WILLIAMS, MORGAN & AMERSON, P.C.			EXAMINER	
10333 RICHMOND AVE.			RUTTEN, JAMES D	
SUITE 1100				
HOUSTON, TX 77042			ART UNIT	PAPER NUMBER
			2192	
			MAIL DATE	DELIVERY MODE
			12/15/2009	PAPER

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

<b>Office Action Summary</b>	<b>Application No.</b>	<b>Applicant(s)</b>	
	10/660,353	BERTELRUD ET AL.	
	<b>Examiner</b>	<b>Art Unit</b>	
	JAMES RUTTEN	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

#### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).

Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) Responsive to communication(s) filed on 31 August 2009.
- 2a) This action is **FINAL**.      2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) Claim(s) 1-4,6-13,15-21 and 23-37 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) Claim(s) \_\_\_\_\_ is/are allowed.
- 6) Claim(s) 1-4,6-13,15-21 and 23-37 is/are rejected.
- 7) Claim(s) \_\_\_\_\_ is/are objected to.
- 8) Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on \_\_\_\_\_ is/are: a) accepted or b) objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).  
a) All    b) Some \* c) None of:
  1. Certified copies of the priority documents have been received.
  2. Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)	4) <input type="checkbox"/> Interview Summary (PTO-413)
2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)	Paper No(s)/Mail Date. _____ .
3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)	5) <input type="checkbox"/> Notice of Informal Patent Application
Paper No(s)/Mail Date _____ .	6) <input type="checkbox"/> Other: _____ .

## **DETAILED ACTION**

1. The reply filed 8/31/09 has been fully considered. Claims 1-4, 6-13, 15-21, 23-37 are pending.

### ***Response to Arguments***

2. Applicant's arguments, see pages 1-4, filed 8/31/09, with respect to the rejection(s) of claim(s) 1-4, 6-8, 16-21, 23-29, and 31-37 under 35 U.S.C. § 102 have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of prior art of record "Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET" by (Robinson et al.).
3. Applicant's arguments, see page 5, filed 8/31/09, with respect to the rejection(s) of claim(s) 33 under 35 U.S.C. § 102 have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of US 6230313 to Callahan, II et al.
4. Applicant's remaining arguments filed 8/31/09 have been fully considered but they are not persuasive as detailed below

On pages 1-3 filed 8/31/09, Applicants essentially argue with respect to claim 9 that prior art of record McKeeman et al. does not teach or disclose compilation in advance of a request from a user to compile the file, and in particular that McKeeman's subsequent and previous compilation cannot both refer to the claimed "initiating compilation" which occurs in advance of a request, and in response to determining that the file has been modified. In other words,

McKeeman teaches at least two compilation events, whereas the claimed invention only calls for one. While this argument is persuasive in light of the limitations of claim 1, the limitations of claim 9 are directed to "instructions that when executed enable a processor to" perform the limitations. These limitations of claim 9 do not require compilation "in advance" of a request as provided in claim 1, and furthermore do not require any particular sequence of operation. As such, Applicants' arguments as developed in light of claim 1 do not generalize to the limitations of claims 9-13 and 15, and are not persuasive for the above reasons.

On page 4 filed 8/31/09, Applicants essentially argue with respect to dependent claims 2-3, 12,-13, 18-21, and 32-35, that prior art of record McKeeman generates source code, not object code. This argument is unpersuasive at least in view of previously cited column 5 lines 23-34 which describes the generation of object code and subsequent executable files. This passage describes the "compile-link-run cycle" which is described at column 4 lines 24-26 as being able to be implemented by the environment of the invention. While later passages may encourage the use of an additional optimizing compiler, this does not diminish the disclosure of object code at column 5 lines 23-34. Therefore, Applicants' argument is not persuasive.

At the bottom of page 4 to the top of page 5 filed 8/31/09, Applicants essentially argue that McKeeman does not teach or disclose the use of markers as claimed in claim 29. However, McKeeman col. 11, lines 44-61 discloses the use of a journal as a "marker" to indicate what portion of a file should be compiled. Applicants have not provided further arguments about why the claimed marker cannot be interpreted according to McKeeman's journal. Therefore, the argument is not persuasive.

***Claim Rejections - 35 USC § 102***

5. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.
6. Claims 9-13 and 15 are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,193,191 (McKeeman et al.).

As per claim 9, *McKeeman et al.* discloses an article comprising one or more machine-readable storage media containing instructions (see, e.g., col. 8, lines 6-39) that when executed enable a processor to:

initiate compiling of a file including one or more code segments (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code);  
detect a user request to compile the file (see, e.g., *Id.* (recompilation is started)); and  
provide a result associated with the compiling in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));  
wherein the instructions when executed enable the processor to initiate compiling of the file based on determining that the file was modified (see, e.g., col. 5, lines 21-23).

As per claim 10, *McKeeman et al.* further discloses the instructions when executed enable the processor to display a message to a user indicating that one or more errors were detected during the compiling (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 11, *McKeeman et al.* further discloses the instructions when executed enable the processor to indicate to a user that the compiling was successful (see, e.g., col. 5, lines 23-34

(errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 12, *McKeeman et al.* further discloses the instructions when executed enable the processor to generate a file containing object code based on compiling the file and to store the object code file in a temporary location (see, e.g., col. 5, lines 30-34).

As per claim 13, *McKeeman et al.* further discloses the instructions when executed enable the processor to move the object code file from the temporary location into a product location based on determining that the compiling of the file was successful and in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 15, *McKeeman et al.* further discloses the instructions that when executed enable the processor to initiate compiling of the file based on determining that the file was modified comprise instructions that when executed enable the processor to indicate in a work queue that the file has been modified and to initiate compiling of the file in response to detecting the indication (see, e.g., col. 11, lines 44-61).

#### ***Claim Rejections - 35 USC § 103***

7. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.
8. Claims 1-4, 6-8, 16-21, 23-29, 31-32, and 34-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,193,191 (*McKeeman et al.*) in view of "Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET" by (Robinson et al.).

As per claim 1, *McKeeman et al.* discloses:

initiating compilation of a file in a processor-based system ... (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

*McKeeman et al.* does not expressly disclose: in advance of a request from a user to compile the file. However, *Robinson et al.* teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of *McKeeman et al.* with the background compiler of *Robinson et al.* in order to provide a timely indication of errors as suggested by *Robinson et al.*.

detecting the user request to compile the file (see, e.g., *Id.* (recompilation is started)); and indicating a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein initiating compilation of the file comprises compiling the file in response to determining that the file has been modified (see, e.g., col. 5, lines 21-23).

As per claim 2, *McKeeman et al.* further discloses initiating compilation of the file comprising compiling the file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 3, *McKeeman et al.* further discloses compiling the file comprising compiling one or more code segments in the file to produce an object code file, and further comprising linking the object code file to produce an executable file (see, e.g., col. 5, lines 37-46).

As per claim 4, *McKeeman et al.* further discloses indicating the status of the compilation of the file comprising at least one of indicating that the compilation was successful and indicating that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 6, *McKeeman et al.* further discloses determining that the file has been modified comprising determining that the modified file has been saved to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 7, *McKeeman et al.* further discloses the file including one or more code segments, wherein initiating compilation of the file in response to determining that the file has been modified comprises:

identifying the modified file in a work queue (see, e.g., col. 11, lines 44-61); and  
initiating the compilation of the file based on the modified file being identified in the work queue (see, e.g., col. 11, lines 44-61).

As per claim 8, *McKeeman et al.* further discloses indicating the status of the compilation of the file comprising generating one or more files associated with the compilation of the file, storing the one or more generated files in a temporary location, and transferring the one or more files from the temporary location to a different location in response to detecting the user request (see, e.g., col. 5, lines 30-34).

As per claim 16, *McKeeman et al.* discloses an apparatus (see, e.g., col. 8, lines 6-39), comprising:

means for initiating compilation of a file in a processor-based system ... (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

*McKeeman* does not expressly disclose: in advance of a request from a user. However, *Robinson et al.* teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of *McKeeman et al.* with the background compiler of *Robinson et al.* in order to provide a timely indication of errors as suggested by *Robinson et al.*.

means for detecting the user request to compile the file (see, e.g., *Id.* (recompilation is started)); and

means for indicating a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the means for initiating compilation initiate compiling the file based on determining that the file was modified (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54).

*McKeeman et al.* further discloses a machine-readable storage media containing instructions for implementing the recited functionality (see, e.g., col. 8, lines 6-39).

As per claim 17, *McKeeman et al.* discloses an apparatus (see, e.g., col. 8, lines 6-39), comprising:

a storage unit having a file stored therein (see, e.g., col. 8, lines 6-39); and

a control unit communicatively coupled to the storage unit (see, e.g., col. 8, lines 6-39), the control unit adapted to:

initiate compilation of the file (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

McKeeman does not expressly disclose: in advance of a request from a user to compile the file. However, *Robinson et al.* teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of *McKeeman et al.* with the background compiler of *Robinson et al.* in order to provide a timely indication of errors as suggested by *Robinson et al.*.

detect the user request to compile the file (see, e.g., *Id.* (recompilation is started)); and

indicate a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported)); wherein the control unit is adapted to compile the file in response to determining that the file has been modified (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54).

As per claim 18, *McKeeman et al.* discloses the control unit is adapted to compile a file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 19, *McKeeman et al.* discloses the control unit is adapted to link the object code file to produce an executable file (see, e.g., col. 5, lines 37-46).

As per claim 20, *McKeeman et al.* discloses the control unit is adapted to store the executable file in a temporary location and to transfer the executable file from the temporary location to a different location based on detecting the user request (see, e.g., col. 5, lines 30-34).

As per claim 21, *McKeeman et al.* discloses the control unit is adapted to at least one of indicate that the compilation was successful and indicate that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 23, *McKeeman et al.* discloses the control adaptation to compile the file in response to determining that the file has been modified comprises:

an adaptation to identify the modified file in a work queue (see, e.g., col. 11, lines 44-61); and

an adaptation to initiate the processing of the file based on the modified file being identified in the work queue (see, e.g., col. 11, lines 44-61).

As per claim 24, *McKeeman et al.* discloses:  
identifying one or more source files that have been modified in a processor-based system (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54);  
initiating processing of at least a portion of the modified source files (see, e.g., col. 11, lines 44-61);

*McKeeman* does not expressly disclose: before receiving a request to process the modified files. However, *Robinson et al.* teaches background compilation in advance of a compilation request (see page 16 "Visual Basic .NET has a compiler that continually works in the background.") It would have been obvious to one of ordinary skill in the art at the time the invention was made to use the compilation of *McKeeman et al.* with the background compiler of *Robinson et al.* in order to provide a timely indication of errors as suggested by *Robinson et al..*

receiving the request to process at least one of the modified files (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code)); and  
providing a status associated with the processing of the file in response to receiving the request (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 25, *McKeeman et al.* further discloses the processor-based system is adapted to execute an integrated development environment module (see, e.g., col. 3, lines 53-56), wherein identifying the one or more files comprises the integrated development environment module placing the one or more of the source files that have been modified in a queue (see, e.g., col. 11, lines 44-61).

As per claim 26, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to a user saving the source file to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 27, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least a portion of one source file in the queue in response to a user saving the source file to a storage unit using an editor and then exiting from the editor (see, e.g., col. 17, lines 11-36).

As per claim 28, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that a user desires to compile at least a portion of the source file as the source file is being edited (see, e.g., col. 5, lines 15-17; col. 11, lines 44-61).

As per claim 29, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that the source file includes at least one marker identifying a section of the source file that should be compiled, and wherein initiating processing of at least the portion of the one or more modified files comprises compiling the identified section of the source file (see, e.g., col. 11, lines 44-61).

As per claim 31, *McKeeman et al.* further discloses initiating the processing comprises initiating a build process to produce a software application and wherein receiving the request comprises receiving the request to building the software application (see, e.g., col. 5, lines 15-17).

As per claim 32, *McKeeman et al.* further discloses initiating the build process comprises performing compiling the modified source files to produce object code files and linking the object code files to produce executable files (see, e.g., col. 5, lines 37-46).

As per claim 34, *McKeeman et al.* further discloses suppressing at least one of an error and warning that is detected while compiling the modified source files (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 35, *McKeeman et al.* further discloses the object code files and the executable files are moved to a different storage location in response to detecting the request and in response to detecting no error or warning (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 36, *McKeeman et al.* further discloses identifying one or more source files comprises identifying the one or more source files based on a directed acyclic graph (see, e.g., col. 16, 54-63).

As per claim 37, *McKeeman et al.* further discloses the directed acyclic graph includes a list of dependent files, wherein identifying one or more source files comprises identifying at least one modified source file and another source file that is dependent on the modified source file using the directed acyclic graph (see, e.g., col. 16, 54-63).

9. Claim 30 is rejected under 35 U.S.C. 103(a) as being unpatentable over *McKeeman et al.* and *Robinson et al.* as applied above, and further in view of U.S. Pat. App. Pub. No. 2005/0108682 (*Piehler et al.*).

As per claim 30, *McKeeman et al.* discloses such a method (see the rejection of claims 24 and 25 under 35 U.S.C. § 102(b)) but fails to expressly disclose initiating the processing of the modified source files comprises causing a background thread to awaken in response to placing the one or more of the source files in the queue, where the background thread thereafter initiates processing of the source files. However, *Piehler et al.* teaches such use of a background thread to enqueue a task for itself to complete the rest of the compilation in a background thread as part of a response to a new file being added to a project or an existing file being modified (*Piehler et al.* at paragraphs [0170] through [0180]. Therefore, it would have been obvious to include such background thread use as per the teachings of *Piehler et al.* One would be motivated to do so to gain the advantage of providing immediate feedback to the user without any detectable visible delay in typifying responsiveness while the compiler finishes processing the change (*Poehler et al.*

at paragraph [0174]).

10. Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over McKeeman et al. and Robinson et al. as applied above, and further in view of U.S. Pat. No. 6,230,313 (Callahan, II et al.).

As per claim 33, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprising placing at least one source file in the queue in response to determining that the source file includes ...[markers] identifying a section of the source file that should be compiled, wherein the ... marker defines the beginning of the portion of the source file and the ... marker defines the end of the portion, and wherein initiating processing of at least the portion of the one or more modified files comprises compiling the identified section of the source file (see, e.g., col. 16, line 60, through col. 17, line 6 (the compiler of *McKeeman* can identify changed portions (having a defined beginning and end) within a source code file and recompile only those changed portions)). *McKeeman* does not expressly disclose: at least two markers. However, *Callahan, II et al.* teaches the use of markers at the beginning and end of a region of interest (see column 10 lines 10-25). It would have been obvious to one of ordinary skill in the art at the time the invention was made to use McKeeman's markers with *Callahan's* teaching of multiple markers in order to easily identify regions of interest as suggested by *Callahan*.

***Conclusion***

11. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Matthew Gertz, "Advanced Basics; Scaling Up: The Very Busy Background Compiler," Microsoft Corp., MSDN Magazine, June 2005, 4 pages, discusses in more detail the implementation of the VB.NET background compiler and the changes in its design from the VISUAL BASIC .NET 2002 development environment to the VISUAL BASIC .NET 2005 development environment.

U.S. 2005/0114771 (Piehler et al.) (claiming the benefit to provisional application 60/449,984, filed February 26, 2003) discloses an editor and compiler integrated with a re-tokenizer that allows incremental compilation of code in the background as the user types it in the editor (paragraphs [0165] through [0176]).

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to JAMES RUTTEN whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 10:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/J. Derek Rutten/  
Examiner, Art Unit 2192